A Presence Service Architecture for Supporting Spontaneous Interaction

Tatsuo Nakajima
Department of Information and Computer Science
Waseda University
tatsuo@dcl.info.waseda.ac.jp

Abstract

Ubiquitous computing environments create a seamless boundary between atoms and bits. Especially, in the near future, embedding various types of sensors in our daily lives enables an application to customize their behavior according to its user's situation. Also, physical objects in our daily lives can be used to manipulate information in cyber spaces. However, the vision requires a new software infrastructure that collaborates various appliances in a spontaneous way.

The paper proposes an architecture to realize the vision in our daily computing environments by creating a society of appliances. Our architecture is called *Pervasive Servers*, that embeds micro servers called *stationary pervasive servers* everywhere. Also, a *personal pervasive server* that is carried by each person coordinates the embedded servers that are near the person. The architecture is very attractive because it is easy to personalize the coordination according to each person's policy. We also show that there are several useful applications based on the architecture.

1 Introduction

Ubiquitous computing[14, 17, 28, 35] has became very popular, and there are many researches that have reported fantastic results every year. However, our daily lives have not been smart yet. For example, many companies have been developed various standard specifications for home computing, but our home appliances could not be connected each other. Also, many researchers have reported the importance of wearable computing, but a person who wears a computer is very rare. Some people define that ubiquitous computing is to embed computers everywhere[1, 5], and sensors connected to the computers make the behavior of applications personalized. However, the definition represents a small part of ubiquitous computing vision. We believe that the following two issues are more important to realize the vision of ubiquitous computing.

- Reducing complexities in our daily lives.
- Focusing on our pleasurable experiences to develop new services.

Our daily lives become more and more complex every day. Information technologies have been increasing the complexities because a large amount of our daily lives is currently spent for analyzing various information[2]. Ironically, current ubiquitous computing technologies will increase the amount of information dramatically, and increase complexities in our daily lives. Also, various appliances surrounding us become commoditize[21]. Today, it is very difficult to create an appliance that has special features. For example, we cannot distinguish each vendor's televisions. Therefore, it is important to take into account pleasurable experiences when a user uses the appliance[25].

In future ubiquitous computing environments, we need to take into account integrating various components from a variety of large-scaled services such as multimedia systems, large-scaled simulation systems based on Grid, and enterprise systems based on EJB to small-scaled services such as sentient objects that change their behavior according to the current situation[8, 36]. The most important issue is that the boundary between atoms and bits should be seamless to reduce the complexities and to increase pleasurable and fantastic experiences in our daily lives. However, a good software infrastructure to collaborate various devices and services is necessary to achieve the above vision. There are several proposals for software infrastructures for ubiquitous computing[6, 7, 11, 12, 18, 19, 32], but a future infrastructure should support that various components are structured or configured spontaneously to make a system self-organizable because it is difficult to assume a priori knowledge about environments, and the entire system will be incrementally extensible. Also, it is important to incorporate legacy devices. The traditional infrastructures have failed to satisfy the requirements.

In this paper, we propose an architecture called *Pervasive Servers* to satisfy the requirements. Each server offers a different service. Our architecture enables these services to be collaborated in a spontaneous way. In our architecture, each person has a *personal pervasive server*. The server will be a part of traditional personal digital assistances(PDA) or mobile cellular phones. Our environments contain micro servers called *stationary personal servers* that can be accessed by the standard HTTP-based protocol. The proposed architecure provides a presence service for supporting spontaneous interaction explicitly. The presence service is similar to a directory service or

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 IEEE

dicovery service, but the service is different from the traditional services because the presence service uses proximity more explicitly, and the service supports context-awareness in a more elegant way.

The remaining of the paper is structured as follows. Section 2 presents issues in ubiquitous computing environments. In Section 3, we show design issues of our architecture. Section 4 shows the overview of our architecture and its prototype implementation. In Section 5, we show four applications of our proposed architecture showing the effectiveness of our approach. Section 6 presents the current status of our system, and Section 7 shows future directions.

2 Ubiquitous Computing Environments

The purpose of a software infrastructure for ubiquitous computing is to retrieve information from our real world, that could not be available before, and to control various everyday objects that could not be controlled before by embedding computers. One of the most important issues in ubiquitous computing is to provide context-awareness to integrate physical and cyber spaces for personalizing our real world to reduce complexities in our daily lives[4]. Many researchers are working on similar research topics such as sentient computing[11], pervasive computing[1], things that think[9], tangible bits[14], affective computing[27], ensemble computing[31], proactive computing[30], and autonomic computing[13].

These researches show that a software infrastructure to support ubiquitous computing is a key to realize its vision. The infrastructure makes it possible to share various devices and sensors, and to build ubiquitous computing applications easily. We have worked with many companies for building embedded systems[21], and found that traditional following choices are wrong.

- A future appliance will be extremely smart.
- Our environment will be extremely smart.
- Each appliance will have a well defined fixed interface.

The first choice means that each application contains many functionalities to satisfy various requirements. For example, current digital televisions are very complex and include various functionalities for satisfying most of us. However, the cost of the television is increased, and it is not easy to extend to support future advanced services. Similarly, the second choice is require very high cost to build smart environments shown in many researches. The most serious problem of the choice is that it is not easy to support the collaboration with environments and appliances that are carried by a user due to special protocols that are assumed by the environments. The last choice cannot allow us to upgrade each appliance's service independently.

A key to solve the problems is to make our environment smart in an incremental and spontaneous fashion by collaborating many specialized appliances and devices. For example, when we like to add a new sensor device, we do not like to modify or replace existing devices. Also, we like to use a new home appliance without reconfiguring the appliance. We believe that the characteristic is very important for an infrastructure for building ubiquitous computing environments practically.

Traditional enterprise systems have adopted distributed objects like CORBA and DCOM to integrate each component. The interface of the component is defined in a fixed way, but the implementation of the component is able to be changed according to future requirements. However, in ubiquitous computing environments, interfaces between components are changed frequently. For example, each vendor likes to change the interface of their appliances to make their products more attractive. We believe that HAVi[10] and Jini[29] have not been used widely due to the reason.

Our proposed architecture makes it possible to maintain environments in a spontaneous and incremental way, but we have adopted simple and standard protocols. Thus, our architecture allows us to incorporate commercial devices and appliances easily. The approach enables us to collaborate various specialized appliances in a spontaneous way to do a user's action, and it is possible to decompose a general appliance into specialized appliances to reduce the complexities caused by the genericity[22].

3 Design Issues

The vision of ubiquitous computing tells us that future appliances will embed computers and networks, but we need a standard way to access the appliances to collaborate them. If each appliance supports a different way to access it, future ubiquitous computing environments will become too complex, and it is very hard to be managed robustly.

Also, many everyday objects will be smart, and it is not impossible to determine their configurations in advance. Therefore, the configurations should be chosen at runtime in a spontaneous way. We believe that the spontaneous approach reduces the complexities in software design dramatically, and traditional constructive approaches are not appropriate to build a large-scaled complex systems for ubiquitous computing.

In our proposed architecture, we have made the following three design choices.

- Each physical object contains a server.
- The server supports a simple and standard protocol.
- Each person has a device that coordinates servers near him.

The first choice means that all appliances that contain computers will execute the same server program although each server has a difference configuration to provide a different functionality. The choice makes our architecture very simple. The second choice makes our server program simple that can be embedded in various appliances.

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 IEEE

Also, it is easy to accommodate commercial appliances that support standard protocols. The third choice makes it possible to coordinate various appliances according to each user's preference. The choice offers two important advantages. The first advantage is that it is easy to protect each user's privacy because each user has a different personal device that contains his private information. The second advantage is that the use to each appliance can be customized easily according to his preference because he carries a configuration profile at any time, and it will be used to configure environments at any places. As described in Section 5.1, the advantage is especially attractive for home computing.

4 Pervasive Server Architecture

Figure 1 shows the pervasive server architecture, and Figure 2 is the current structure of a pervasive server. Our architecture consists of two components. The first component is a *stationary pervasive server*, and the second component is a *personal pervasive server*. A stationary pervasive server is contained in various appliances and daily objects in our environments to offer functions for collaborating with other appliances and objects. A personal pervasive server is carried by each person to configure stationary pervasive servers. The personal pervasive server can be embedded in PDAs, cellular phones, or personal servers proposed in [33]. In our current implementation, pervasive servers support HTTP-based protocols such as UPnP[34].

The stationary pervasive servers announce their presences periodically. A personal pervasive server detects the announcements, and retrieve service specification from the stationary pervasive servers to know their attributes and functionalities. The personal pervasive server also announces its presence in order to find a display near the device to show user interface, and forwards commands to control detected stationary pervasive servers. Also, the announcement from the personal pervasive server can be used to collaborate with other personal pervasive servers to support spontaneous interpersonal interaction.

In our approach, we assume that a pervasive server architecture supports three types of embedded servers. The first type of servers is a UPnP-based appliances that can be controlled via the SOAP protocol. In the near future, many UPnP-based home appliances will be available in our homes, and our system can provide better personalization and context-awareness to support home computing. The second type is a traditional Web server that provides various information. For example, each place provides its information by a stationary pervasive server located in the place. Similarly, various information can be provided by traditional Web servers, and various Web servers providing information in respective environments should be installed surrounding us in the near future[5]. The last type is a server supporting the XML messaging protocol. The protocol simply transmits a message containing a XML document. For example, a server containing a sensor returns a XML document that contains a retrieved value from the sensor.

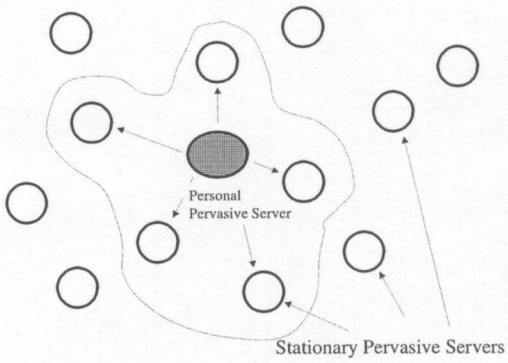

Figure 1: Structure of Pervasive Server Architecture

In our current implementation, the services can be accessed via the SOAP protocol or the HTTP protocol, and its presence can be known via the SSDP protocol. Our system has adopted a component framework based on the OSGi gateway service specification[24]. The personal pervasive server has a service database to store service specifications about detected pervasive servers. Finally, a UI component generates a presentation document to control servers or to show their information, and transmits it to a display device near a user.

In our system, each server stores a service specification document that contains several attributes and operations that can be accepted. The attributes are specified in an RDF(Resource Description Framework) document, and the operations are represented in a WSDL(Web Service Description Language) document. If a server offers the same services, the WSDL document may be provided by a company that defines the service. A personal pervasive server collects the service specification documents of detected servers, and stores the information in its service database.

As described above, in our architecture, many servers are embedded in our daily lives, and a personal pervasive server integrates services provided by the servers. Each application has a different way to integrate services. For example, a command to an appliance can be forwarded to a target server in a context-aware way in a personal home server application. Also, the collected information can be used to compose several services in a seamless town application.

5 Example Applications

In this section, we present four example applications to show the effectiveness of our approach. The first one is *Personal Home Server*, the second is *Proximity Search Engine*, the third is *People Presence Manager*, and the final one is *Seamless Town*.

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 **IEEE**

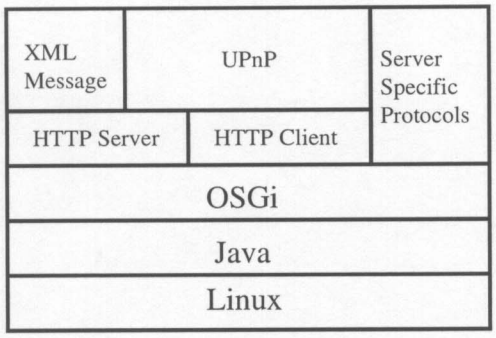

Figure 2: Structure of Pervasive Server

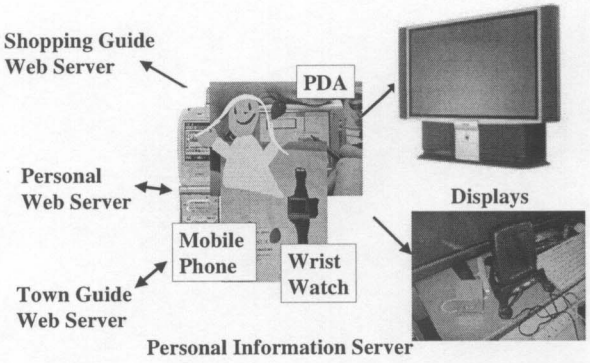

Figure 4: Proximity Search Engine

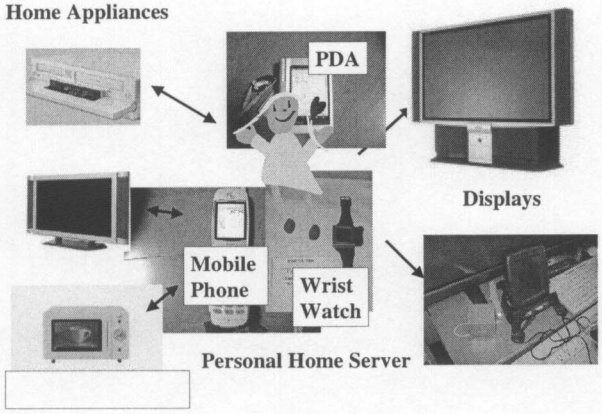

Figure 3: Personal Home Server

5.1 Personal Home Servers

Currently, many consumer companies have been working on developing home networks. They like to connect respective home appliances to control them in an integrated fashion. In a home network, a home gateway is a key component to coordinate the home appliances in respective homes. However, a home gateway has serious problems in ubiquitous computing environments. For example, a user cannot control the same type of appliances in a similar way when he visits to his friend's house because each home gateway may have different policy, and the policy is dedicated to a place not a person. A personal home server[20] will solve many problems of the home gateway.

As shown in Figure 3, each person has a personal home server that is a personal pervasive server in our architecture. The server detects home appliances near the person, that are stationary personal server in our architecture, and retrieves the information about the appliances. Also, a list of the appliances is shown on the nearest display. In our current implementation, when a display detects a personal home server, it retrieves a HTML document from the personal home server. The HTML document automatically generated by the server from the information stored in its database contains URLs to control respective appliances. In the future implementation, we have a plan to embed the URLs to control appliances in various presentation documents such as MS Power Point, Flash, and SMIL documents.

The approach is very attractive to support future ubiquitous computing environments because a user's personal home server to control them in different spaces. Also, it is easy to customize the control according to each user's preference. The most important advantage of our architecture is that our system can use traditional UPnP based home appliances, and they do not require special supports to be controlled from personal home servers.

5.2 Proximity Search Engine

In the near future, the complexities in our daily lives will be increased, and we need a better support to get information near us. The proximity search engine provides a mechanism to offer necessary information when we visit to various places. For example, we like to know persons who have the same interests near us. In this case, we assume that each person has a personal Web server, and he carries it at any time. The server is embedded in a personal device such as a cellular phone.

Figure 4 shows how to use the proximity search engine. The server detects Web servers near a person who has the proximity search engine. It accesses the detected Web servers, and retrieves HTML documents from the servers. They may be personal Web servers carried by other persons, maintained by shops, managed by towns, or containing various sensors. The user interface for searching necessary information can be navigated on a display near the user similar to a personal home server. The contents shown on the display contain various information

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 IEEE

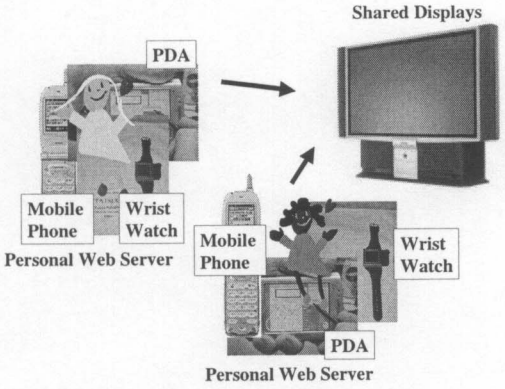

Figure 5: People Presence Manager

surrounding us.

The system can also be used to compose several storage systems in a spontaneous way. For example, we assume that a user has music storages in a hi-fi audio system, in a public server on the Internet and his portable MP3 player. In his house, these storages are integrated spontaneously, and he can search his favorite song from these storages. Each music has metadata containing information that are used for searching. Currently, we are considering audio and video streams, photos and HTML documents for searching information. There are many approaches to represent metadata such as RDF and MPEG7. We are currently investigating which metadata format is appropriate to collect information in a spontaneous way.

5.3 People Presence Manager

The third application is a people presence manager that shows a user's personal Web documents near a display. Thus, a user can know who is near him. Also, he can examine information about a person near him. For example, in our research labolatory, we can know who is in our labolatory now. Also, if the personal Web server monitors persons who have the personal servers, a user can know the status of each person by checking his Web documents.

As shown in Figure 5, a display device detects personal Web servers near the display. The display device can show Web documents for each person. A user can know who is near the display, and their detailed information by accessing the Web documents.

There are many ways to show the presence of people. One approach is to show the number of person near a display. The display may show a small amount of information such as his interests, or his current status without violating his privacy by filtering information in Web documents. Also, we may use an ambient display[36] to show the presence of people.

The application can be extended in several ways. The first extension is to support spontaneous communication among people. Each personal server returns information about a person who is in the same public space. Organizing each person's information in a spontaneous way is to allow us to browse information about people near us, and to contact the person to chat something.

The second extension is to show the presence of objects. The information is useful to build context-aware applications. Also, the presence manager can be used to find an object that a user lost in an easy way.

5.4 Seamless Town

The last application that we are designing is *Seamless Town*. The application offers a new way supporting the interaction with various services in a town. Usually, a user needs different ways to interact services according to a user's situation. For example, when a user is on a street, he may use his cellular phone, but when he is in a shop, he will get information from a kiosk terminal in the shop. The approach confuses a user because he needs to know which method is available at each place. The pervasive server architecture offers a very flexible approach to interact services. On a street, a user tries to find a forwarding service that forwards a SOAP request to an actual service on the Internet to get information about the street. He can see the information on a display near him. On the other hand, in a shop, he may access the shop's Web server, and see the information on a display in the shop. The user is always able to use the same interaction technique anytime anywhere. We call the technique *universal interaction*.

However, we need to consider the following two issues to realize universal interaction in a town. The first issue is how to integrate various services seamlessly. In a town, various services will be offered, and it is not easy to know which service is useful for him. Therefore, it is important to compose low level services to build a high level service that is preferable for the user. The approach can reduce the number of services that can be seen from the user. We believe that the most important issue to design ubiquitous computing environments is to reduce the complexities in our real world because the most scarce resource is a user's attention in the near future.

The second issue is how to offer context information for composing services. The composition according to a user's preference and situation needs various context information about our daily lives. The first step to retrieve the information requires various sensors. However, current sensor technologies are not suitable to retrieve our context information. We believe that future materials will contain many small computers and sensors, and they give various information to us[3, 15]. For example, they calculate geographic relationship among materials, or provide unique identifiers that are useful to know what they are. We call the intelligent materials *sentient materials*. The real sentient materials may not be available soon, but we like to consider the impact of the sentient materials to ubiquitous computing while building the seamless town applications. Also, it is important to create a large-scaled database to store information about our real world. The database stores abstract information extracted from many sentient materials in our world. We are considering to adopt a peer-to-peer technology such as Pastry and Chord to build the scalable context information database.

6 Current Status and Discussions

The section shows the current status of our system, and presents some discussions while designing the system.

6.1 Current Status

We have implemented a simple prototype of our architecture, and are building applications describe in the paper. Our system has been written in Java, and we adopted Linux as an underlying operating system. Stationary pervasive servers have run on standard personal computers or small embedded computers. Also, a personal pervasive server has adopted a Compaq's iPAQ PDA. and it can be communicated with other servers by using IEEE 802.11b.

Currently, our current implementation has adopted the OSGi framework[24]. The OSGi framework allows us to decompose software into components, and it makes it possible to load/unload components dynamically. Therefore, when a personal pervasive server requires a new application component for coordinating servers, the component can be loaded from close servers. The characteristic is very attractive for our proposed architecture. Also, since the OSGi framework is defined as a standard specification, we will use many commercial components for building future applications.

6.2 Discussions

This section describes five topics to build pervasive server architecture. The first topic is about the size of our system. Currently, our system is written in Java, and runs on Linux. The infrastructure requires 32bit processors and at least 4 MBytes memory. However, a small processor that executes Java byte codes is available now, and Linux runs on even a cellular phone. Therefore, the size is not a problem to implement our pervasive servers, but the cost is still expensive to embed the servers in various everyday objects.

The second topic is a communication infrastructure of the pervasive server architecture. The servers can use any network technologies such as Bluetooth, IEEE802.11, ZigBee, or Infrared. Since the current implementation assume to use the IP protocol to communicate among servers. Now, the IP protocol is ubiquitous, and various appliances start to support the protocol, but it may not be a good choice to be used as an underlying network protocol, especially when we need to take into account battery consumption. We need to investigate what kind of a network infrastructure is suitable for the pervasive server architecture in the near future.

The third topic is how to integrate Web services in our architecture. Web services are very useful to access various services on the Internet. In the near future, various services such as on-line shopping and air ticket reservation can be accessed via the SOAP protocol. Also, the computational resources on the Internet will be virtualized by using the technology[23]. The technology will make the Internet a distributed operating system. Thus, it is desirable to access various services in ubiquitous computing environments via the SOAP protocol. We believe that the protocol will be useful to integrate ubiquitous computing and Internet computing seamlessly, and integrate various services both in ubiquitous computing and on the Internet.

The fourth topic is how to support context-awareness in our architecture. Our system mainly uses proximity as context information to integrate services in a spontaneous way. Our approach is very easy to be implemented, but the integration is based on the range within that servers can communicate. For example, in a office we like to use services in the room, but in a town, we need to access services on a street where he walks. We believe that we should gain experiences of users whether we need more expensive approaches to integrate services, or the current simple spontaneous approach is enough.

The last topic is how to structure our system. Currently, we are using OSGi to structure the systems. In the approach, each function can be implemented as a component, and a system configures several components for respective services. When a system needs a new service, it can download the service from the Internet. However, in the future, the configuration of a system should be changed according to the current situation, and a system should behave in a predictable way. This means that a newly downloaded component should satisfy constraints of underlying platforms. Currently, we are working on a new component framework called the *autonomic component framework*, In the framework, each component specifies a protocol to define the abstract behavior of the component. The protocol is represented as a state machine, and the state transition describes constraints of the transitions. The constraints are used to check whether the component can satisfy the requirements of a system before running it. The state machine can also show the dependencies among components and assumptions of an underlying platform. Also, a meta protocol specifies tacit dependencies among components. For example, a component that contains threads needs to say which scheduling policy is assumed to execute the component.

A component in the framework needs to be designed to make recovery time short because in ultra heterogeneous environments, it is very hard to build a service that does not fail. We think that an approach proposed in [26] is promising to build reliable systems for ubiquitous computing. Our component framework will provide a mechanism to support the recovery-oriented approach. Also, information received from stationary pervasive servers and stored in a personal pervasive server should be soft states. The states should be automatically removed when the personal pervasive server cannot contact with the stationary pervasive servers. We believe that the approach can reduce complexities of our system, and decrease the cost of future software development.

7 Future Work

In the near future, we like to consider the following five issues. The first issue is whether proximity is enough to coordinate servers. Our architecture focus on information about proximity to integrate appliances and devices. However, integrating Web services requires to know what kind of services on the Internet. In our architecture, we

assume that a stationary pervasive server that forwards requests to the Web services, and returns information about available Web services to a personal pervasive server. If the server returns information about all Web services on the Internet, the personal pervasive server cannot manage the information of the services. The stationary pervasive server should filter what kind of services in which a user has interests.

The next issue is how to compose appliances. It is not easy to compose appliances spontaneously because there are many combinations of appliances and it is not easy to determine which combination is valid. We think that we need to specify a service composition specification to integrate several servers. Let us assume that a user likes to use a television that has a VCR functionality. A personal pervasive server for the user has a composition specification for the appliance. When the server detects that a TV tuner and a VCR device are currently available, it sends commands to connect the functions. Each personal pervasive server contains several composition specifications, and a display near him shows a list of appliances that can be used by him currently. However, we need to consider how the approach changes our daily lives significantly. For example, a vendor can sell a new appliance as a new composition specification. When a user buys the specification from an on-line shop and downloads it, he can use the functions offered by appliances near him to satisfy the specification. If his personal pervasive server cannot detect a function to compose a new appliance to satisfy the specification, he may need to buy a server that contains the missing function. The approach may create a new business model for consumer electronics.

Our system adopts the XML-based approach to communicate among respective servers. As described in previous sections, future appliances should be evolved independently. An advantage of the document-based approach is that each server can interpret a document independently. Each XML document contains several tags, and each personal pervasive server determines how to interpret the tags in different ways. For example, some tags may be ignored or interpreted in different ways. In the near future, we like to investigate the feature in our application examples. Especially, in the seamless town application, each server is deployed and replaced independently. Thus, we believe that the approach is very useful.

We are considering to organize a society of sentient objects that is context-aware daily objects. We believe that sentient objects are very important because each person likes to have personalized objects for him. However, if every object independently changes its behavior in a context-aware way, the behavior may be inconsistent according to each user's mental model[16]. This is a reason why we need to take into account a society of sentient objects. Sentient objects in a society should be self-organized to behave consistently.

Finally, in our projects, we found that it is important to represent semantic information about each person and object in a standard way. The information is called *ontology*. The ontology is a specification of a conceptualization. In our systems, we need to define various ontologies such as ontologies for context information, for information on Web, and for information about persons and objects. There are several approaches to define ontologies, but the definition is complex and inappropriate for ubiquitous computing. In the future, we need to investigate how to support ontologies for ubiquitous computing.

8 Conclusions

The paper has proposed a new architecture to build ubiquitous computing environments. Our architecture assumes that every appliance and daily object contains a server called a stationary pervasive server. Also, a personal pervasive server that is carried by each person coordinates stationary pervasive servers, and provides a personalized way to access the servers.

Currently, we are working to develop four example applications to show the effectiveness of our architecture. The first one is *personal home server*, the second is *proximity search engine server*, the third is *people presence manager*, and the last one is *seamless town*. The applications can be useful for building ubiquitous computing environments, and their implementations become simple by using the pervasive server architecture.

References

[1] G.Banavar, J.Beck, E.Gluzberg, J.Munson, J.Sussman, D.Zukowski, "Challenges: An Application Model for Pervasive Computing", In Proceedings of the Six Annual International Conference on Mobile Computing and Networking, 2000.

[2] J.S. Brown, P. Duguid, "The Social Life of Information", Harvard Business School Press, 2000.

[3] W. Butera, "Programming a Paintable Computer", MIT PH.D Thesis, 2002.

[4] W.Buxton, "Less is More(More or Less)", In Invisible Computing, P.J. Denning(Ed.), 2002.

[5] Gaetano Borriello and Roy Want, "Embedded Computation Meets the World-Wide-Web", Communications of the ACM, Vol. 43 No.5. 2000.

[6] A.Dey, G.D. Abowd, D.Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction, Vol.16, No.2-4, 2001.

[7] W.K.Edwards, N.W.Newman, J.Sedivy, T.Smith, S.Izadi, "Challenge: Recombianant Computing and the Speackeasy Approach", Mobicom 2001, 2001.

[8] H.-W.Gellersen. A.Schmidt, and M.Beigl, "Adding Some Smartness tp Devices and Everyday Things", In Proceedings of the Third Workshop on Mobile Computing System and Applications, 2000.

[9] N.Gershenfeld, "When Things Start to Think", Owl Book, 2000.

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 **IEEE**

[10] HAVi Consortium, "HAVi Specification", http://www.havi.org.

[11] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, Paul Webster, "The Anatomy of a Context-Aware Application", In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999.

[12] J.I.Hong, J.A.Landay, "An Infrastructure Approach to Context-Aware Computing", Human-Computer Interaction, Vol.16, No.2-4, 2001.

[13] P.Horn, "Autonomic Computing", IBM Manifesto, http://www.research.ibm.com/autonomic/manaifesto/autonomic_computing.pdf, 2001.

[14] H. Ishii, B.Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", In Proceedings of Conference on Human Factors in Computing Systems,1997.

[15] L.H.Lifton, "Pushpin Computing: A Platform for Distributed Sensor Networks", MIT Master Thesis, 2002.

[16] C.Lueg, "On the Gap Between Vision and Feasibility", In Proceedings of Pervasive 2002, 2002.

[17] T. Nakajima, et. al., "Technology Challenges for Building Internet-Scale Ubiquitous Computing", In Proceedings of the Seventh IEEE International Workshop on Object-oriented Real-time Dependable Systems, 2002.

[18] T.Nakajima, D.Ueno, I.Satoh, H.Aizu, "A Virtual Overlay Network for Integrating Home Appliances", In the Proceedings of the 2nd International Symposium on Applications and the Internet, 2002.

[19] T.Nakajima, "Experiences with Building Middleware for Audio and Visual Networked Home Appliances on Commodity Software", In Proceedings of ACM Multimedia 2002, 2002.

[20] T. Nakajima, I. Satoh, "Personal Home Server: Enabling Personalized and Mobile Home Computing Environments", In Preparation.

[21] T.Nakajima, "Implication of Embedded Linux in Japanese Embedded Industries", In Proceedings of International Symposium on Object-Oriented Real-Time Distributed Computing, 2003.

[22] D.A. Norman, "The Invisible Computer: Why Good Products can fail, the personal computer is so complex, and information appliances are the solution", MIT Press, 1998.

[23] Open Grid Services Architecture Working Group, "Open Grid Services Architecture", http://www.ggf.org/ogsa-wg/.

[24] Open Server Gateway Initiative, "OSGi Service Platform Specification", http://www.oegi.org/.

[25] B.J.Pine II, J.H. Gilmore, "The Experience Economy", High Bridge Company, 1999.

[26] D.A. Perterson, et.al., "Receovery Oriented Computing(ROC): Motivation, Definition, Techniques, and Case Studies, UC Berkeley, Technical Report UCB/CSD-02-1175, 2002.

[27] R. Picard, "Affective Computing", The MIT Press, 1997.

[28] K.Raatikainen, H.B.Christensen, T.Nakajima, "Applications Requirements for Middleware for Mobile and Pervasive Systems", ACM Mobile Computing and Communications Review, Vol.16, No.4, 2002.

[29] Sun Microsystems, "Jini Technology", http://www.jini.org/

[30] D.L. Tennenhouse, "Proactive Computing", Communication of the ACM, Vol.43, No.5, 2000.

[31] P.Thomas, H.W. Gellersen, "Ensemble Computing", International Journal of Human-Computer Interaction, Vol.13, No.2, 2001.

[32] D.Ueno,, T.Nakajima, I.Satoh, K.Soejima, "Web-based Middleware for Home Entertainment", In Proceedings of International Conference on ASIEN'02, 2002.

[33] R.Want, T.Pering, G.Danneels, M.Kumar,M.Sundar, J.Light, "The Personal Server: Changing the Way We Think About Ubiquitous Computing", In Proceedings of Ubicomp2002.

[34] Universal Plug and Play Forum, "Universal Plug and Play Specification", http://www.upnp.org/.

[35] Mark Weiser, "The Computer for the 21st Century", Scientific American, Vol. 265, No.3, 1991.

[36] G. Wisneski, H. Ishii, et. al. "Ambient Display: Turning Architecture Space into an Interface between People and Digital Information", In Proceedings of the First International Workshop on Cooperative Buildings, 1998.

Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03)
0-7695-2054-5/04 $20.00 © 2004 **IEEE**